BAB II
TINJAUAN PUSTAKA

2.1 Model *Waterfall*

Model *waterfall* (*waterfall model*) yang juga disebut sebagai *Linear Sequential Model* merupakan model yang sistematik, pendekatan-pendekatan berurutan untuk membangun perangkat lunak yang dimulai dengan persyaratan dan *progress* spesifikasi pelanggan (*communication*) melalui perencanaan (*planning*), pemodelan (*modeling*), konstruksi (*construction*), dan penyebaran (*deployment*) [2].

Model *waterfall* memiliki tahapan-tahapan sebagai berikut [2]:

1. *Communication* (*Project Initiation*, *Requirements Gathering*)

   Sebelum pekerjaan teknis dapat dimulai, sangat penting untuk berkomunikasi dan berkolaborasi dengan pelanggan atau para pemangku kepentingan lainnya (*stakeholders*). Hal tersebut dimaksudkan untuk memahami tujuan s*takeholder* dari proyek dan mengumpulkan keperluan untuk membantu mendefinisikan fitur dan fungsi dari perangkat lunak.

2. *Planning* (*Estimating, Scheduling*, *Tracking*)

   Aktivitas perencanaan membuat sebuah peta (*map*) yang dapat membantu memandu tim dalam proyek. Peta tersebut mendefinisikan rekayasa perangkat lunak dengan menjelaskan tugas-tugas teknis yang dilakukan, risiko-risiko yang mungkin terjadi, sumber daya yang diperlukan, produk kerja yang mau dibuat, dan penjadwalan kerja.

3. *Modeling* (*Analysis*, *Design*)

   Pembuatan rancangan untuk memahami gambaran besar (seperti arsitektur, bagaimana unsur-unsur tersebut cocok satu sama lain, dan karakteristik lainnya). Bila diperlukan, rancangan dibuat menjadi semakin detil agar pemahaman terhadap masalah semakin besar dan bagaimana masalah tersebut dapat diselesaikan.

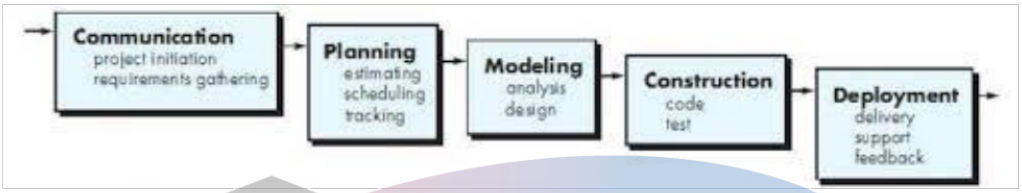
4. *Construction* (*Code*, *Test*)

   Sesuatu yang dirancang harus dibangun. Aktivitas tersebut menggabungkan pembuatan kode dan melakukan pengujian yang diperlukan untuk mengatasi *error* pada kode.

5. *Deployment* (*Delivery*, *Support*, *Feedback*)

   Perangkat lunak akan disebarkan kepada pelanggan yang akan mengevaluasi produk dan memberikan *feedback* berdasarkan evaluasi.

   Gambar berikut ini menunjukkan tahapan-tahapan dari model *waterfall* [2].



Gambar 2.1 Model *Waterfall*

2.2 Teknik Pengembangan Aplikasi

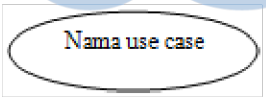
2.2.1 *Use Case Diagram*

Use case diagram atau *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case diagram* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dalam sistem informasi yang akan dibuat. Secara kasar, *use case diagram* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu [3].

Berikut ini merupakan simbol-simbol yang ada pada *use case diagram* [3]:

1. *Use Case*

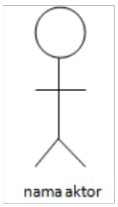
   Use case merupakan fungsionalitas yang disediakan oleh sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor. *Use case* dinyatakan dengan menggunakan kata kerja (*verb*) di awal frase nama *use case*.



Gambar 2.2 Simbol *Use Case*

2. Aktor (*Actor*)

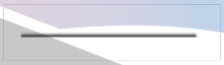
   Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.



Gambar 2.3 Simbol Aktor

3. Asosiasi (*Association*)

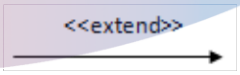
   Asosiasi merupakan komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.



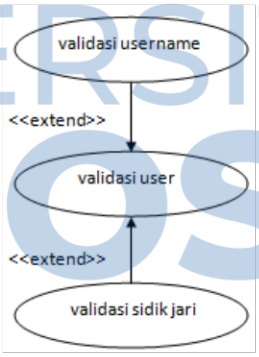
Gambar 2.4 Simbol Asosiasi

4. Ekstensi (*Extend*)

   Ekstensi merupakan relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu. *Use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.



Gambar 2.5 Simbol Ekstensi (*Extend*)



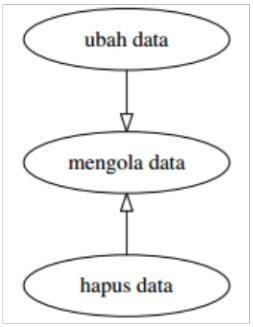
Gambar 2.6 Contoh Penggunaan Ekstensi (*Extend*)

Pada gambar di atas, arah panah mengarah pada *use case* yang ditambahkan.

5. Generalisasi (*Generalization*)

   Generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.

Gambar 2.7 Simbol Generalisasi (*Generalization*)



Gambar 2.8 Contoh Penggunaan Generalisasi (*Generalization*)

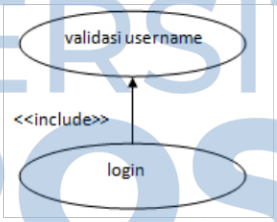
6. Menggunakan atau *Include* (*Use*)

   Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau syarat dijalankan *use case* ini.

Gambar 2.9 Simbol *Include* atau *Uses*

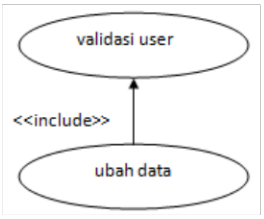
Terdapat dua sudut pandang yang cukup besar mengenai *include* di *use case*, yaitu:

1. *Include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan, misalnya pada kasus berikut ini.



Gambar 2.10 Contoh Penggunaan *Include* I

2. *Include* berarti *use case* yang ditambahkan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* yang ditambahkan telah dijalankan, misalnya pada kasus berikut ini.



Gambar 2.11 Contoh Penggunaan *Include* II

Kedua interpretasi di atas dapat dianut salah satu atau keduanya, tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

2.2.2 *Entity Relationship Diagram* (ERD)

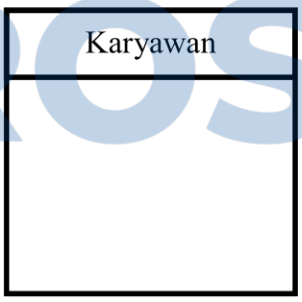
ERD merupakan representasi grafis dari model E-R. Model E-R merupakan representasi logis dan terperinci dari data untuk suatu organisasi atau untuk area bisnis. ERD dinyatakan dalam bentuk entitas dalam lingkungan bisnis, hubungan atau asosiasi di antara entitas, dan atribut atau properti dari kedua entitas dan hubungannya [4].

Berikut ini merupakan komponendasar dari ERD, antara lain [4, 5]:

1. Entitas (*Entity*)

   Entitas merupakan seseorang, tempat, objek, peristiwa, atau konsep dalam lingkungan pengguna tentang bagaimana organisasi mengelola data. Dengan demikian, sebuah entitas mempunyai nama kata benda. Berikutini merupakan contoh jenis entitas:

   a. Orang: karyawan, murid, pasien
   b. Tempat: toko, gudang, negara
   c. Objek: mesin, gedung, mobil
   d. Peristiwa: penjualan, registrasi, pembaharuan
   e. Konsep: akun, mata kuliah, pusat kerja

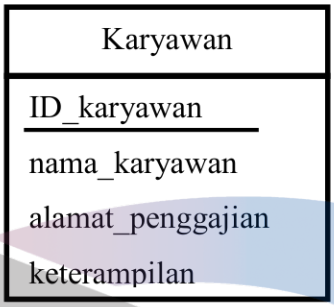


Gambar 2.12 Entitas (*Entity*)

2. Atribut

   Setiap entitas mempunyai sekumpulan atribut di dalamnya. Sebuah atribut merupakan sebuah properti atau karakteristik dari entitas. Dengan demikian, sebuah atribut mempunyai nama kata benda. Berikut ini merupakan contoh atribut dari entitas.

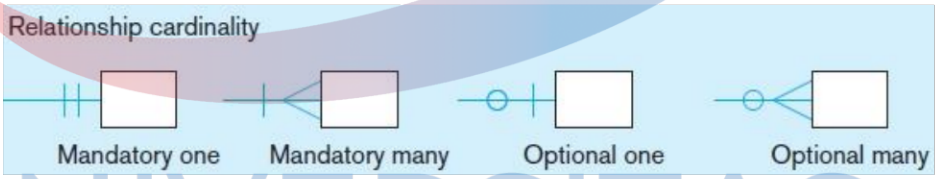
a. Murid: ID murid, nama murid, alamat rumah, nomor telepon, mata pelajaran
b. Mobil: ID kendaraan, warna, berat, daya
c. Karyawan: ID karyawan, nama karyawan, alamat penggajian, keterampilan



Gambar 2.13 Atribut

3. Hubungan Kardinalitas (*Cardinality Relationship*)

Sebuah basis data yang distruktur dengan baik membangun hubungan (*relationship*) antara entitas. Hubungan kardinalitas menunjukkan seberapa banyak entitas suatu jenis berhubungan dengan seberapa banyak entitas jenis lainnya.



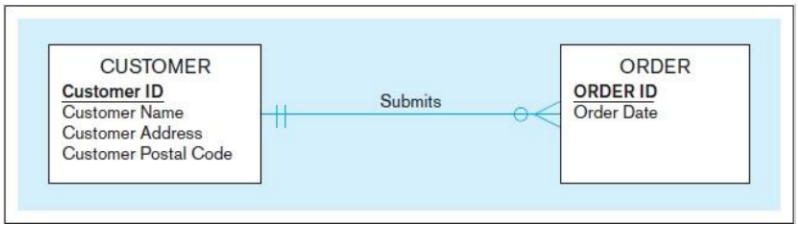
Gambar 2.14 Hubungan Kardinalitas (Cardinality *Relationship*)

4. Hubungan Binary (*BinaryRelationship*)

Hubungan Binary (*Binary Relationship*) merupakan hubungan antara dua entitas:

1. Hubungan Satu ke Banyak (*One-to-Many*)

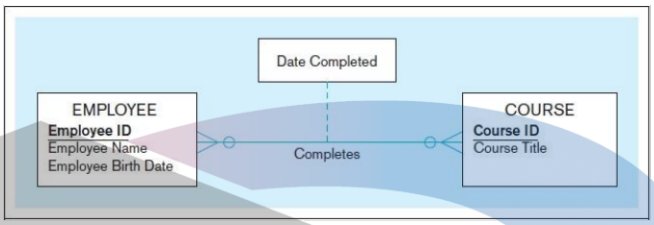
Satu entitas A dapat berhubungan lebih dari satu dengan entitas B. Namun tidak berlaku sebaliknya, dimana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.



Gambar 2.15 Hubungan Satu ke Banyak (*One-to-Many*)

2. Hubungan Banyak ke Banyak (*Many-to-Many*)

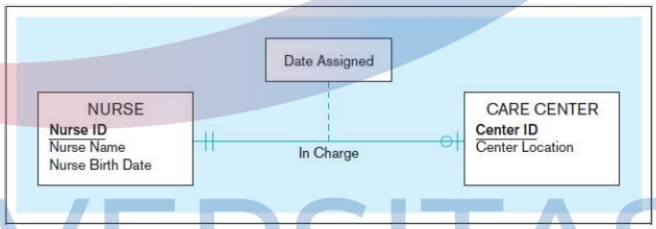
   Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.



Gambar 2.16 Hubungan Banyak ke Banyak (*Many-to-Many*)

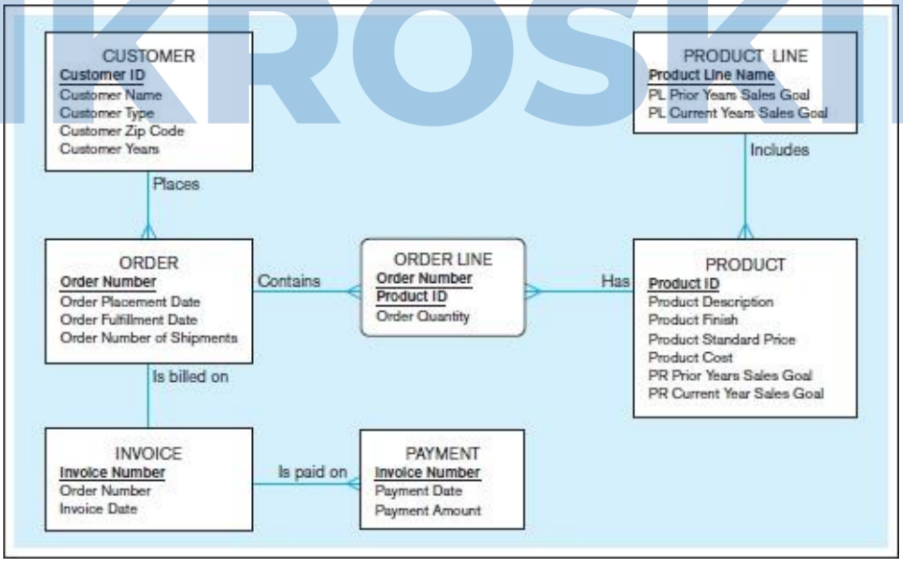
3. Hubungan Satu ke Satu (*One-to-One*)

   Satu entitas A berhubungan paling banyak satu dengan entitas B. Begitupunsebaliknya, satu entitas pada himpunan B berhubungan paling banyak satu dengan entitas A.



Gambar 2.17 Hubungan Satu ke Satu (*One-to-One*)

Berikut ini adalah contoh penggambaran ERD [4].



Gambar 2.18 Contoh Penggambaran ERD

2.3 Sistem Basis Data

Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan. Pada intinya basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat [3].

Komponen-komponen utama dalam sebuah basis data terdiri dari Perangkat Keras (*Hardware*), Sistem Operasi (*Operating System*), Basis Data (*Database*), Sistem Aplikasi Pengelola Basis Data (DBMS), dan Pemakai (*User*) [5, 3]:

1. Perangkat Keras (*Hardware*)

   Perangkat keras yang biasanya terdapat dalam sebuah sistem basis data adalah sebagai berikut:

   a. Komputer (satu untuk sistem yang *stand-alone* atau lebih dari satu untuk sistem jaringan)
   b. Memori sekunder yang *online* (*harddisk)*
   c. Memori sekunder yang *offline* (*removable disk)* untuk keperluan *backup* data
   d. Media atau perangkat komunikasi (untuk sistem jaringan)

2. Sistem Operasi (*Operating System*)

   Sistem operasi merupakan program yang mengaktifkan sistem komputer, mengendalikan seluruh sumber daya atau *resource* dalam komputer, dan melakukan operasi-operasi dasar dalam komputer. Sejumlah sistem operasi yang banyak digunakan seperti MS-DOS, MS-Windows, Linux (untuk komputer *stand-alone* atau untuk komputer *client* dalam sistem jaringan) atau Novel-Netware, MS-Windows Server, Unix, Linux (untuk komputer *server* dalam sistem jaringan komputer). Program pengelola basis data hanya dapat aktif atau berjalan bila sistem operasi yang dikehendakinya atau sesuai telah aktif.

3. Basis Data (*Database*)

   Basis data atau *database* terdiri dari dua kata, yaitu basis dan data. Basis dapat diartikan sebagai markas, gudang, atau tempat berkumpul, sedangkan data merupakan representasi fakta dunia nyata yang mewakili suatu objek, seperti manusia, barang, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang diwujudkan dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau

kombinasinya. Basis data dapat didefinisikan menjadi beberapa dalam sejumlah sudut pandang, seperti:

a. Himpunan kelompok data atau arsip yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dangan cepat dan mudah.
b. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan atau redundansi yang tidak perlu untuk memenuhi berbagai kebutuhan.
c. Kumpulan *file* atau tabel atau arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronik.

Sebuah sistem basis data dapat memiliki beberapa basis data. Setiap basis data berisi sejumlah objek basis data, seperti tabel. Di samping berisi data, setiap basis data juga menyimpan definisi struktur, baik untuk basis data maupun objek-objeknya secara rinci.

4. Sistem Aplikasi Pengelola Basis Data (DBMS)

Pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah perangkat lunak atau sistem yang khusus. Perangkat lunak yang dimaksud adalah *Database Management System* (DBMS) yang akan menentukan bagaimana data diorganisasi, disimpan, diubah, dan diambil kembali. Selain itu, DBMS juga menerapkan mekanisme pengamanan data, pemakaian data secara bersama, pemaksaan keakuratan atau konsistensi data, dan sebagainya. Suatu sistem aplikasi disebut sebagai DBMS bila memenuhi persyaratan sebagai berikut:

a. Menyediakan fasilitas untuk mengelola akses data.
b. Mampu menangani integritas data.
c. Mampu menangani akses data yang dilakukan.
d. Mampu menangani *backup* data.

DBMS versi komersial yang paling banyak digunakan di dunia saat ini yaitu Oracle, Microsoft SQL Server, IBM DB2, dan Microsoft Access, sedangkan DBMS versi *open source* yang cukup berkembang dan paling banyak digunakan yaitu MySQL, PostgreSQL, Firebird, dan SQLite.

5. Pemakai (*User*)

   Berdasarkan cara berinteraksi, terdapat beberapa jenis atau tipe pemakai suatu sistem basis data, yaitu:

   a. *Programmer* Aplikasi

      Pemakai yang berinteraksi dengan basis data melalui *Data Manipulation Language* (DML) yang disertakan dalam program yang ditulis dalam bahasa pemrograman induk (C, C++, Pascal, PHP, Java, dan sebagainya).

   b. Pengguna Mahir (*Casual User*)

      Pemakai yang berinteraksi dengan sistem tanpa menulis modul program, dimana menggunakan *query* (untuk mengakses data) dengan bahasa *query* yang telah disediakan sebelumnya.

   c. Pengguna Umum (*End User* atau *Naive User*)

      Pemakai yang berinteraksi dengan sistem basis data melalui pemanggilan satu program aplikasi permanen (*executable program*).

   d. Pengguna Khusus (*Specialized User*)

      Pemakai yang menulis aplikasi basis data nonkonvensional dan untuk keperluan-keperluan khusus, seperti untuk aplikasi *Artificial Intelligence*, Sistem Pakar, Pengolahan Citra, dan sebagainya yang bisa mengakses basis data dan/atau tanpa DBMS yang bersangkutan.

2.4 Aplikasi Mobil

Aplikasi mobil merupakan aplikasi yang berjalan dalam perangkat mobil yang mudah digunakan dan dapat diakses dimanapun dan kapanpun. Aplikasi mobil mencakup sekumpulan program perangkat lunak yang berjalan dalam perangkat mobil dan melakukan tugas tertentu. Aplikasi mobil mudah digunakan, *user friendly*, tidak mahal, serta dapat diunduh dan dijalankan di hampir seluruh ponsel [6].

Terdapat tiga jenis aplikasi mobil, yaitu [7, 8]:

1. Aplikasi *native* adalah aplikasi yang dikembangkan untuk perangkat tertentu. Contohnya, bahasa pemrograman Java untuk aplikasi Android dan bahasa pemrograman Objective-C atau Swift untuk aplikasi iPhone.
2. Aplikasi *hybrid* adalah gabungan dari aplikasi *native* dan aplikasi *web*. Biasanya aplikasi *hybrid* menggunakan *browser* untuk mengizinkan aplikasi *web*mengakses

berbagai fitur di perangkat mobil seperti *Push Notification*, *Contacts*, atau *Offline Data Storage*. Beberapa alat untuk mengembangan aplikasi ini adalah Phonegap atau Rubymotion.

3. Aplikasi *web* adalah aplikasi yang digunakan perusahaan, dimana situs *web* pada dasarnya memiliki kemiripan dengan aplikasi mobil. Aplikasi *web* ditulis dengan bahasa pemrograman HTML5 atau Javascript dan berjalan dengan *browser* ganda, seperti Safari atau Chrome.

2.5 Android

Android merupakan sebuah sistem operasi berbasis Linux yang didesain khusus untuk perangkat bergerak, seperti *smartphone* atau *tablet*. Android menyediakan *platform* yang terbuka (*open sources*) bagi para pengembang untuk menciptakan aplikasi maupun memodifikasi sistem operasi [9]. Oleh sebab itu, saat ini bila dibandingkan dengan sistem operasi yang lain untuk perangkat *handphone* dan PC, Android mempunyai dukungan aplikasi dan *game* nonberbayar terbanyak yang bisa diunduh oleh penggunanya melalui Google *Play*.

Android secara sederhana bisa diartikan sebagai sebuah perangkat lunak yang menggunakan basis kode komputer yang bisa digunakan pada perangkat mobil yang mencakup sistem operasi, *middleware* (perangkat lunak yang berperan sebagai "penengah" antara sebuah aplikasi dengan aplikasi lain untuk mempermudah proses integrasi antara aplikasi-aplikasi tersebut), serta aplikasi kunci yang dirilis oleh Google. Oleh sebab itu, Android mencakup keseluruhan aplikasi, mulai dari sistem operasi sampai pada pengembangan aplikasi itu sendiri.

Dalam pengembangan, aplikasi Android menyediakan Android SDK yang menyediakan *tools* dan API untuk para pengembang aplikasi dengan *platform* Android. Android menggunakan dasar bahasa pemrograman Java, yaitu kode Java yang terkompilasi bersama-sama dengan data dan *file resources* yang dibutuhkan oleh aplikasiyang digabungkan oleh *aapt tools* menjadi paket Android, sebuah *file* yang ditandai dengan *suffixapk*. *File* ini didistribusikan sebagai aplikasi dan diinstalasi pada perangkat mobil. Tetapi secara sempit, Android biasanya mengacu pada sistem operasinya saja [10].

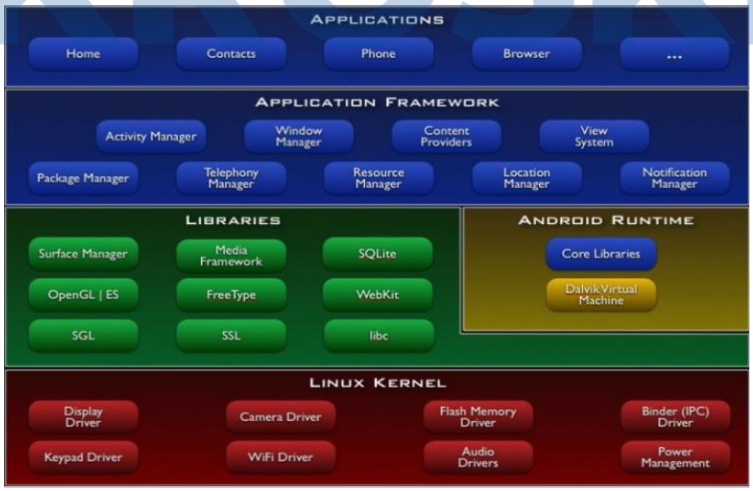
2.5.1 Fitur-Fitur Android

Berikut ini merupakan beberapa fitur Android, yaitu [11]:

1. *Storage*, menyediakan *database* SQLite sebagai sebuah *database* relasional untuk penyimpanan data.
2. *Connectivity*, mendukung komunkasi GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, *Bluetooth*, Wi-Fi, LTE, dan WiMax.
3. *Messaging*, mendukung MMS dan SMS.
4. *Web browser*, dengan menggunakan *web kit* dan Chrome yang tersedia didalamnya.
5. Multimedia, mendukung beragam format data multimedia seperti 3GP, MP3, MP4, JPEG, PNG, GIF, BMP, dan sebagainya.
6. *Hardware*, mendukung penggunaan perangkat keras eksternal sebagai sensor, seperti *camera*, *digital compass*, GPS, dan sebagainya.
7. *Multi-touch*, mendukung *multi touch* layar sentuh.
8. *Multi-tasking*, mendukung pengembangan aplikasi *multi-tasking*.
9. *Thetering*, mendukung koneksi internet yang berfungsi sebagai *wired/wireless hotspot*.

2.5.2 Arsitektur Android

Android merupakan kernel Linux yang menyediakan dan mengatur alur proses aplikasi. Gambar di bawah ini merupakan struktur dari sistem operasi Android [11].



Gambar 2.19 Arsitektur Android

Google mengibaratkan Android sebagai sebuah tumpukan *software*. Setiap lapisan dari tumpukan ini menghimpun beberapa program yang mendukung fungsi-fungsi spesifik dari sistem operasi. Berikut ini adalah lima susunan dari lapisan-lapisan tersebut jika dilihat dari lapisan teratas hingga lapisan dasar [11]:

1. *Application*

   Application merupakan bagian yang memuat aplikasi-aplikasi yang dapat digunakan oleh pengguna perangkat Android. Pada bagian ini, Andorid menyediakan fungsi-fungsi dasar *smartphone* seperti menelepon dan mengirim pesan singkat, menjalankan *web browser*, mengakses daftar kontak, dan lain-lain. Bagi rata-rata pengguna, lapisan inilah yang paling sering diakses. Fungsi-fungsi dasar tersebut diakses melalui *user interface*. Selain aplikasi inti seperti yang terdapat pada arsitektur Android, aplikasi-aplikasi tambahan yang diinstalasi sendiri oleh penguna juga akan menempati bagian *Application* dan memiliki hak akses yang sama terhadap *Application Framework.*

2. *Application Framework*

   Lapisan selanjutnya adalah *application framework*, yang mencakup program untuk mengatur fungsi-fungsi dasar *smartphone*. *Application Framework* merupakan bagian yang dapat digunakan oleh pengembang aplikasi untuk membuat aplikasi Android dengan menggunakan serangkaian *tool* dasar, seperti alokasi *resourcesmartphone*, aplikasi telepon, pergantian antar proses atau program, dan pelacakan lokasi fisik telepon. Android menawarkan kemampuan pengembangan untuk membangun aplikasi yang sangat kaya dan inovatif. *Programmer* mendapatkan akses penuh untuk memanfaatkan *Android Protocol Interface* (API) yang juga digunakan *core applications*dan dapat diakses oleh aplikasi-aplikasi inti dari Android. Di dalam semua aplikasi terdapat servis dan sistem yang meliputi:

   a. Satu set *Views* yang dapat digunakan untuk membangun aplikasi yang meliputi *lists, grids, text boxes, buttons*, dan *embeddable* web *browser*.
   b. *Content Providers* yang memungkinkan aplikasi untuk mengakses data dari aplikasi lain (misalnya *Contacts*), atau untuk membagi data yang dimilikinya.
   c. *Resource Manager*, menyediakan akses ke *non-code resources*, misalnya *localized strings, graphics*, dan *layout files.*

d. *Notification Manager* yang memungkinkan semua aplikasi untuk menampilkan *custom alerts* pada *status bar.*

e. *Activity Manager* yang mengatur*life cycle* dari aplikasi dan menyediakan *common navigation backstack*.

3. *Libraries*

Bertempat di level yang sama dengan Android *Runtime* adalah *Libraries.* Android menyertakan satu set *library-library* dalam bahasa C/C++ yang digunakan oleh berbagai komponen yang ada pada sistem Android. Kemampuan ini dapat diakses oleh pengembang melalui Android *Application Framework*. Sebagai contoh, Android mendukung pemutaran format audio, video, dan gambar. Berikut ini beberapa *core library* tersebut:

a. *System C library,* diturunkan dari implementasi standar C *system library* (libc) milik BSD, dioptimasi untuk piranti *embedded* berbasis Linux.

b. *Media Libraries*, berdasarkan OpenCORE *PacketVideo*; *library* ini mendukung *playback* dan *recording* dari berbagai format audio dan video populer, meliputi MPEG4, H.264, MP3, AAC, AMR, JPG, dan PNG.

c. *Surface Manager*, mengatur akses pada *display* dan lapisan komposisigrafis 2D and 3D dari berbagai aplikasi.

d. *LibWebCore*, *web browser* yang mendukung Android *browser* maupun *embeddable web view*.

e. *Scalable Graphics Library*(SGL), *library* yang menangani pengelolaan grafis 2D.

f. 3D *libraries*, *library* yang menangani pengelolaan grafis 3D.

g. *FreeType*, *library* yang menangani pengelolaan *rendering font*.

h. SQLite, digunakan untuk pengelolaan *database*.

4. Android *Runtime*

Lapisan setelah *Library* adalah Android *Runtime*. Android *Runtime* ini berisi *Core Libraries* dan Dalvik *Virtual Machine*. *Core Libraries* mencakup serangkaian inti *library* Java, artinya Android menyertakan satu *setlibrary* dasar yang menyediakan sebagian besar fungsi-fungsi yang ada pada *library* dalam dasar bahasa pemrograman Java. Dalvik *Virtual Machine* (DVM) merupakan *virtual machine* yang digunakan untuk menerjemahkan instruksi-instruksi program Java ke dalam

instruksi yang dimengerti oleh sistem operasi. Namun, dalam *platform* Android, *virtual machine* yang digunakan bukan Java *Virtual Machine* (JVM), tetapi Dalvik *Virtual Machine* (DVM). Dalvik VM adalah sebuah *virtual machine* yang dioptimasi untuk perangkat yang memiliki memori kecil, sumber daya terbatas, dan kemampuan proses yang kecil. Dalvik VM mengeksekusi *file* dalam bentuk Dalvik *executable* (.dex). Dalvik telah dibuat sehingga sebuah piranti yang memakainya dapat menjalankan multi *virtual machine* dengan efisien. Dalvik *Virtual Machine* (VM) menggunakan kernel Linux untuk menjalankan fungsi-fungsi seperti *threading* dan *low-level memory management*.

5. Linux Kernel

   Tumpukan paling bawah pada arsitektur Android adalah kernel. Google menggunakan kernel Linux versi 2.6 untuk membangun sistem Android, yang mencakup *memory management*, *security setting*, *power management*, dan beberapa *driver hardware*. Kernel berperan sebagai *abstraction layer* antara *hardware* dan keseluruhan *software*. Sebagai contoh, HTC GI dilengkapi dengan kamera. Kernel Android terdapat *driver* kamera yang memungkinkan pengguna mengirimkan perintah kepada *hardware* kamera.

2.5.3 Pengembangan Aplikasi Android

Ada 4 (empat) hal mendasar yang harus dipahami dalam membangun aplikasi berbasis Android [11]:

1. *Activity*, adalah tampilan grafis yang dilihat ketika menjalankan sebuah aplikasi. Aplikasi dapat memiliki lebih dari satu *Activity*.
2. *Intent*, adalah serangkaian nilai yang menunjukkan apa yang harus dilakukan ketika terjadi perpindahan layar.
3. *Service*, adalah layanan yang bekerja di belakang layar (*background*).
4. *Content provider*, memungkinkan sebuah aplikasi untuk dapat menyimpan dan menerima data dari *database*.

2.5.4 Android SDK

Android SDK mencakup *tools* pengembangan yang komprehensif. Android SDK terdiri dari *platform*, *debugger*, *libraries*, *handset emulator*, dokumentasi, contoh

kode program, dan tutorial yang diperlukan untuk mengembangkan aplikasi Android. Android SDK dibangun sebagai *add-on* untuk pembangunan JavaKit dan memiliki *plugin* terintegrasi untuk Eclipse *Integrated Development Environment*. IDE yang didukung secara resmi adalah Eclipse 3.2 atau lebih dengan menggunakan *plugin* Android *Development Tools* (ADT), dengan ini pengembang dapat menggunakan IDE untuk mengedit dokumen Java dan XML serta menggunakan peralatan *command line* untuk menciptakan, membangun, melakukan *debug* aplikasi Android, dan pengendalian perangkat Android (misalnya *reboot*, menginstalasi paket perangkat lunak) [12].

2.5.5 Android SDK *Debug Tools*

Android SDK menyediakan beberapa *tools* untuk digunakan dalam *debugging*, seperti Android *Debug Bridge*, *LogCat*, *Hierarchy Viewer*, dan *TraceView*. *Tools* tersebut dapat ditemukan di alat/direktori instalasi Android SDK [12].

2.6 Teknologi Pengembangan Aplikasi Mobil

2.6.1 *Global Positioning System*(GPS)

GPS adalah sistem satelit navigasi dan penentuan posisi yang dimiliki dan dikelola oleh Amerika Serikat. Sistem ini didesain untuk memberikan posisi dan kecepatan tiga dimensi serta informasi mengenai waktu secara kontinu diseluruh dunia. Saat ini GPS sudah banyak digunakan manusia diseluruh dunia dalam berbagai bidang aplikasi yang menuntut informasi, seperti dapat memberikan informasi tentang posisi, kecepatan, dan waktu secara cepat, akurat, murah, dan dimana saja dibumi ini tanpa bergantung pada cuaca. Hal tersebut perlu dicatat bahwa GPS merupakan satu-satunya sistem navigasi ataupun sistem penentu posisi dalam beberapa abad ini.

Sistem operasi Android dapat dikombinasikan dengan radio GPS sehingga memudahkan pengembang membuat aplikasi yang dapat mendeteksi lokasi pengguna pada suatu waktu. Selain itu, GPS juga dapat mendeteksi pergerakan atau perubahan lokasi pengguna ketika berpindah dari suatu lokasi ke lokasi lainnya [13].

2.6.2 Google *Maps* API

Google *Maps* adalah layanan gratis yang diberikan oleh Google dan sangat populer di kalangan masyarakat. Google *Maps* adalah suatu peta dunia yang dapat digunakan untuk melihat suatu daerah. Dengan kata lain, Google *Maps* merupakan suatu peta yang dapat dilihat dengan menggunakan *browser*. Cara membuat Google *Maps* untuk ditampilkan pada suatu situs *web* atau *blog* sangat mudah, hanya dengan pengetahuan mengenai HTML serta JavaScript, serta koneksi internet yang sangat stabil. Dengan menggunakan Google *Maps* API dapat menghemat waktu dan biaya serta dapat menambahkan fitur Google *Maps* dalam situs *web* yang telah dibuat atau pada *blog* yang berbayar maupun gratis sekalipun dengan Google *Maps* API [13].

Google *Maps* API adalah suatu *library* yang berbentuk JavaScript untuk membangun aplikasi peta digital yang handal sehingga dapat fokus hanya pada data yang akan ditampilkan. Dengan kata lain, aplikasi Android yang dibuat dapat diintegrasikan dengan Google API. Google telah menyediakan Android *Map* API yang dapat digunakan secara langsung dalam aplikasi, seperti mendeteksi lokasi pengguna aplikasi. Fitur tersebut tentu akan menghemat waktu dibandingkan harus membuatnya dari awal [13].

Pada Google *Maps* API, terdapat empat jenis pilihan model peta yang disediakan oleh Google, diantaranya adalah [13]:

1. *Roadmap*, untuk menampilkan peta biasa berbentuk dua dimensi.
2. *Satelite*, untuk menampilkan foto satelit.
3. *Terrain*, untuk menunjukkan relief fisik permukaan bumi dan menunjukkan seberapa tingginya suatu lokasi, contohnya gunung dan sungai.
4. *Hybrid*, untuk menunjukkan foto satelit yang diatasnya terdpat gambar yang tampil pada *Roadmap* (jalan dan nama kota).

Pada pembuatan program Google *Maps* APIterdapat langkah-langkah yang perlu diikuti pada umumnya, yaitu sebagai berikut [13]:

1. Memasukan *Maps* API JavaScript ke dalam HTML.
2. Membuat element div dengan nama map_canvas untuk menampilkan peta.
3. Membuat beberapa objek literal untuk properti-properti pada peta.
4. Menulis fungsi JavaScript untuk membuat objek peta.
5. Menginisiasi peta dalam elemen body HTML dengan *eventonload*.

2.6.3 MySQL

MySQL adalah salah satu teknologi basis data yang dikembangkan oleh Oracle dan sering digunakan dalam sistem komputasi awan (*cloud computing*). MySQL merupakan *open source Relational Database Management System* (RDBMS) atau sistem manajemen basis data relasional yang berjalan sebagai *server*dan menyediakan akses pengguna lebih dari satu (*multi-user access*) ke sejumlah basis data [14].Sebuah basis data relasional menyimpan data dalam relasi yang disebut dengan tabel. Setiap tabel terdiri dari *record*, dan atribut atau *field*. Setiap *record* di dalam tabel diidentifikasi dengan sebuah atribut atau *field* unik [15].

2.6.4 Node.js

Pada awalnya JavaScript hanya digunakan untuk membuat situs *web* saja. Namun saat ini, berkat adanya Node.js, para pengembang dapat membuat aplikasi *desktop*, situs *web*, maupun perangkat mobil, dan sebagainya. Node.js diciptakan oleh Ryan Dahl pada tahun 2009. Pada awalnya, bahasa pemrograman JavaScript hanya dapat berjalan di atas *browser* karena terdapat *runtime engine* di dalamnya. *Runtime engine* merupakan mesin yang dibutuhkan oleh aplikasi untuk berjalan di komputer. Oleh sebab itu, munculnya Node.js yang mengeluarkan *engine* yang terdapat dalam *browser* sehingga JavaScript dapat diekseskusi di luar *browser*. Node.js merupakan sebuah *platform* untuk mengeksekusi program JavaScript di luar *browser*. Node.js menggunakan *runtime engine* bernama V8 yang merupakan JavaScript *runtime engine* dari Google Chrome [16].

Keuntungan menggunakan Node.js yaitu sebagai berikut [17]:

1. Node.js menggunakan bahasa pemrograman JavaScript yang diklaim sebagai bahasa pemrograman yang paling populer dan banyak dikenal oleh masyarakat luas.
2. Node.js mampu menangani ribuan koneksi bersamaan dengan penggunaan sumber daya minimum untuk setiap prosesnya.
3. Node.js sangat diandalkan terutama untuk membuat aplikasi *real-time*.
4. Node.js adalah proyek *open source* sehingga siapapun dapat melihat struktur kode dan dapat berkontribusi untuk mengembangkannya.

5. Penggunaan JavaScript di sisi *client* dan *sever* meminimalisir ketidakcocokan antar kedua sisi lingkungan pemrograman, seperti komunikasi data yang mana yang menggunakan struktur JSON yang sama di kedua sisi, validasi *form* yang sama yang dapat dijalankan di kedua sisi tersebut, dan sebagainya.
6. *Database* NoSQL, seperti MongoDB dan CouchDB mendukung langsung JavaScript sehingga antarmuka dengan *database* akan lebih mudah.
7. Node.js memakai V8 yang selalu mengikuti perkembangan standar ECMAScript sehingga tidak perlu khawatir bahwa *browser* tidak dapat mendukung fitur-fitur Node.js.

Node.js merupakan *server open source* yang gratis dan menggunakan bahasa pemrograman JavaScript yang berjalan pada *platform* Windows, Linux, Unix, Mac OS X, dan sebagainya [18]. Tugas umum sebuah *web server* yaitu dapat membuka *file* dalam *server* dan mengembalikan konten kepada *client*. Oleh sebab itu, Node.js digunakan untuk menanggani permintaan *file* dan caranya adalah sebagai berikut [18]:

1. Mengirim tugas ke sistem *file* komputer.
2. Bersiap untuk menangani permintaan selanjutnya.
3. Ketika sistem *file* telah dibuka dan membaca *file*, *server* akan mengembalikan konten kepada *client*.

Adanya Node.js akan menghemat memori karena Node.js merupakan *single-threaded*, *non-blocking*, dan *asynchronouslyprogramming*. Selain itu, Node.js dapat mengenerasi konten halaman secara dinamis, dapat membuat, membuka, membaca, menulis, menghapus, dan menutup *file* dalam *server*, dapat mengumpulkan formulir data, dan dapat menambah, menghapus, dan mengubah data dalam *database* [18].

2.6.5 React Native

React Native merupakan sebuah kerangka kerja (*framework*) JavaScript yang dikembangkan oleh Facebook untuk membuat aplikasi iOS dan Android. React Native berbeda dengan *framework* lainnya karena pada *framework* lain memerlukan *browser* untuk menjalankan aplikasi, sedangkan pada React Native berjalan secara *native* (asli) di perangkat iOS dan Android tanpa menggunakan *browser* [19].

Para pengembang dapat menggunakan React Native untuk membangun aplikasi mobil lintas *platform* (*cross-platform*) menggunakan JavaScript [20]. Lintas

platform artinya sebuah perangkat lunak dapat digunakan di sistem operasi yang berbeda.

Keuntungan menggunakan React Native [21]:

1. Aplikasi mobil dapat dibangun dengan bantuan teknologi *website* dimana para pengembang dapat secara mudah menambah keterampilan dan membuat aplikasi React.
2. React Native membantu membangun aplikasi mobil yang *cross-platform*.
3. Menghemat waktu dan biaya untuk membangun aplikasi mobil lebih dari satu *platform*.
4. Blok bangunan yang digunakan pada aplikasi iOS dan Android juga dapat digunakan saat membuat aplikasi berbasis React Native yang berarti React Native adalah kerangka kerja mobil yang menyusun komponen aplikasi untuk aplikasi mobil asli di JavaScript.
5. Sebuah aplikasi React Native menjamin kecepatan dan kelincahan aplikasi mobil dengan responsif dan pengalaman pengguna aplikasi *native* yang baik.

2.6.6 Otentikasi

Otentikasi merupakan teknik untuk pembuktian terhadap identitas suatu entitas, baik orang, kartu kredit, atau mesin. Teknik ini merupakan proses dimana satu pihak dapat mejamin atau memastikan pihak kedua terlibat langsung. Terdapat tiga basis pada teknik otentikasi, yaitu [22]:

1. Sesuatu yang diketahui (*something known*), yaitu otentikasi berdasarkan sesuatu yang diketahui dan dapat diingat, seperti *password* dan *Personal Identification Number* (PIN).
2. Sesuatu yang dimiliki (*something possesed*), yaitu otentikasi berdasarkan pada sesuatu yang dimiliki, seperti *token* dan *smartcard*.
3. Sesuatu yang melekat (*something inheret*), yaitu otentikasi dengan menggunakan bagian unik anggota tubuh, seperti sidik jari, iris mata, dan bagian tubuh unik lainnya.

2.6.7 Firebase

Firebase adalah sebuah *platform database* buatan Google yang mempermudah para pengembang aplikasi. Awalnya pada tahun 2011, Firebase didirikan oleh Andrew Lee dan James Tamplin dengan nama perusahaan Envolve. *Realtime database* merupakan salah satu layanan yang pertama kali dikembangkan oleh mereka. Oleh sebab itu, Google menganggap layanan tersebut sangat bagus dan memiliki potensial sehingga pada tahun 2014, Google mengakuisisi Firebase. Pada tahun 2016, Google memperkenalkan Firebase di acara tahunan Google I/O. Layanan Firebase terus dikembangkan oleh Google dan layanan tersebut cukup banyak digunakan oleh para pengembang aplikasi Android, iOS, maupun *web* [23].Sebuah aplikasi yang membutuhkan otentikasi atau *cloudmessaging* untuk notifikasi dapat menggunakan Firebase. Hal tersebut karena Firebase menyediakan layanan secara terpisah yang dapat digunakan langsung oleh para pengembang. Selain itu, para pengembang tidak perlu kebingungan dengan penskalaan, kinerja *server*, dan ukuran basis data karena Firebase menskala semuanya secara otomatis [24].

2.6.7.1 Otentikasi Firebase

Firebase memiliki fitur *authentication* yang berfungsi sebagai penyimpanan data pengguna (*user*). Pada saat ini, sebagian besar aplikasi ingin mengetahui identitas penggunanya sehingga nanti aplikasi dapat menyimpan data pengguna secara aman di *cloud* dan memberikan pengalaman personal yang sama di setiap perangkat pengguna. Fitur ini menyediakan layanan *backend* dengan SDK yang mudah dan siap digunakan untuk mengotentikasi pengguna ke aplikasi. Oleh sebab itu, dengan menggunakan fitur ini, para pengembang dapat membuat *login* menggunakan GMail, Facebook, Twitter, dan sebagainya [23].

Berikut ini adalah fitur-fitur otentikasi Firebase:

1. Penyedia otentikasi yang luas, baik dengan *email* atau *password*, nomor telepon, Facebook, Google, Twitter, GitHub, Anonymous, dan bahkan dapat terintegrasi dengan penyedia otentikasi khusus yang didukung.
2. Dapat dengan mudah menghubungkan (*link*) akun dari beberapa penyedia.
3. Terdapat SDK untuk aplikasi *web*, iOS, Android, dan C++.
4. Didukung oleh Firebase sehingga akan menskala penyimpanan tanpa batas.

5. Berintegrasi baik dengan fungsi *cloud* untuk Firebase sehingga dapat mengirim *email* kepada pengguna yang mendaftar, bahkan tanpa menggunakan *server*.
6. Fiturnya gratis (kecuali SMS yang digunakan untuk otentikasi via telepon).

2.6.7.2 Firebase *Cloud Messaging* (FCM)

FCM merupakan sebuah fitur yang berfungsi untuk pengiriman pesan lintas *platform* yang memungkinkan seseorang mengirimkan pesan dengan terpercaya dan tanpa biaya sepersenpun [23].

Berikut ini merupakan kemampuan yang dimiliki, antara lain [25]:

1. FCM dapat mengirimkan pesan notifikasi yang akan ditampilkan kepada pengguna atau pesan data dan menentukan sepenuhnya apa yang terjadi dalam kode aplikasi.
2. FCM dapat mendistribusikan pesan ke aplikasi klien, baik dengan cara mendistribusikannya ke sebuah perangkat, ke grup perangkat, atau ke perangkat yang berlangganan topik.
3. FCM dapat mengirim pesan dari aplikasi klien (mengirim notifikasi, *chat*, dan pesan lain dari perangkat ke *server* melalui saluran koneksi FCM yang andal dan hemat baterai).

Implementasi FCM mencakup dua komponen utama untuk mengirim dan menerima pesan [25]:

1. Lingkungan terpercaya, seperti *Cloud Functions for* Firebase atau *server* aplikasi yang akan digunakan untuk membuat, menargetkan, dan mengirim pesan.
2. Aplikasi klien iOS, Android, atau *web* (JavaScript) yang menerima pesan.

Terdapat alur dalam implementasi FCM, yaitu [25]:

1. Menyiapkan FCM SDK

   Menyiapkan Firebase dan FCM pada aplikasi sesuai petunjuk penyiapan untuk *platform*.
2. Mengembangkan Aplikasi Klien

   Menambah penanganan pesan, logika langganan topik, atau fitur opsional lain ke aplikasi klien. Selama pengembangan, pengguna dapat mengirim pesan pengujian dengan mudah dari *Notifications Composer*.

3. Mengembangkan *Server* Aplikasi

   Menentukan apakah para pengembang ingin menggunakan Firebase Admin SDK atau salah satu protokol *server* untuk membuat logika pengiriman, yaitu logika untuk mengautentikasi, membuat permintaan pengiriman, menangani respon, dan sebagainya. Kemudian, buat logika di lingkungan terpercaya. Perlu diperhatikan bahwa jika ingin menggunakan pengiriman pesan *upstream* dari aplikasi klien, harus menggunakan XMPP, dan bahwa *Cloud Functions* tidak mendukung koneksi tetap yang diperlukan oleh XMPP.

2.6.8 *OneSignal*

OneSignal merupakan sebuah *platform* penyedia layanan *push notification*. *OneSignal* memiliki reabilitas (keandalan) yang baik, secara konsisten mempertahankan *uptime* di atas 99,95% untuk pengiriman notifikasi. *OneSignal* memiliki API yang lebih sederhana dalam pengiriman notifikasi dan sudah mendukung fitur seperti ekspor data, pelacakan analitik secara terperinci, penyaringan (*filter*) data secara *real-time*, dan fitur lainnya. *OneSignal* perlu menggunakan FCM (*Firebase Cloud Messaging*) API karena semua aplikasi dalam *Google Play Android* harus menggunakan *Firebase* API untuk mengirimkan notifikasi (FCM) [26].

2.6.9 *Quick Response* (QR)*Code*

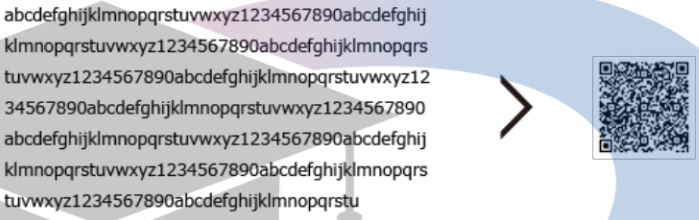
QR *Code* adalah gambar dua dimensi yang merepresentasikan suatu data, terutama data berbentuk teks. QR *Code* merupakan evolusi dari *barcode* yang awalnya satu dimensi menjadi dua dimensi. QR *Code* memiliki kemampuan menyimpan data yang jauh lebih besar daripada *barcode*. Selain itu, QR *Code* digunakan sebagai salah satu metode pengamanan data [27].

Gambar 2.20 Contoh QR *Code*

QR *Code* memiliki enam fitur sebagai berikut [28]:

1. Pengkodean Data Kapasitas Tinggi

   Ketika *barcodes* konvensional hanya memiliki kemampuan untuk menyimpan maksimal 20 digit, QR *Code* memiliki kemampuan menyimpan berlusin sampai beratus informasi. QR *Code* dapat menangani semua tipe data, seperti numerik dan karakter alfabet, kanji, kana, hiragana, simbol, *binary*, dan pengendalian kode. Lebih dari 7.089 karakter dapat dikodekan dalam satu simbol.



Gambar 2.21 Jumlah Data dalam Sebuah QR *Code*

Pada gambar di atas dapat dilihat bahwa dalam sebuah QR *Code* seukuran tersebut dapat mengkode 300 alfanumerik.

2. Memiliki Ukuran Cetak yang Kecil

   Karena QR *Code* membawa informasi baik secara horizontal maupun vertikal, QR *Code* mampu mengkodekan jumlah data yang sama sekitar sepersepuluh ruang dari *barcode* tradisional. QR *Code* memiliki ukuran mulai dari yang besar, kecil, bahkan ukuran mikro sekalipun.

Gambar 2.22 Perbandingan Ukuran *Barcode* dengan QR *Code*

3. Kemampuan Mengkode Kanji dan Kana

   Sebagai simbologi yang dikembangkan di Jepang, QR *Code* mampu mengkodekan sekelompok karakter kanji Level 1 dan Level 2 JIS.Dalam kasus bahasa Jepang, satu karakter Kana atau Kanji yang panjang dapat dikodekan dalam 13 bit sehingga memungkinkan QR *Code* untuk menyimpan lebih dari 20% data daripada simbologi 2D lainnya.

Gambar 2.23 Kana dan Kanji Dikodekan Menjadi QR *Code*

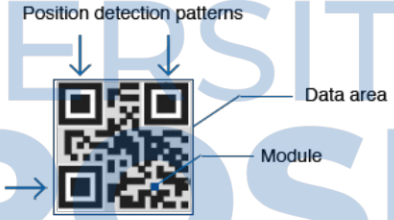
4. Memiliki Ketahanan Terhadap Kotoran dan Kerusakan

   QR *Code* memiliki kemampuan memperbaiki *error*. Data dapat dikembalikan apabila sebagian simbol menjadi kotor atau rusak. Maksimum 30% *codework* (sebuah *codework* merupakan suatu unit yang membangun area data, pada QR *Code*, satu *codework* sama dengan 8 bit) dapat dikembalikan (pemulihan data tidak dapat dikembalikan seutuhnya berdasarkan besarnya kerusakan dan kotor).

Gambar 2.24 QR *Code* yang Kotor dan Jorok

5. Dapat Dibaca dari Berbagai Arah dalam 360º

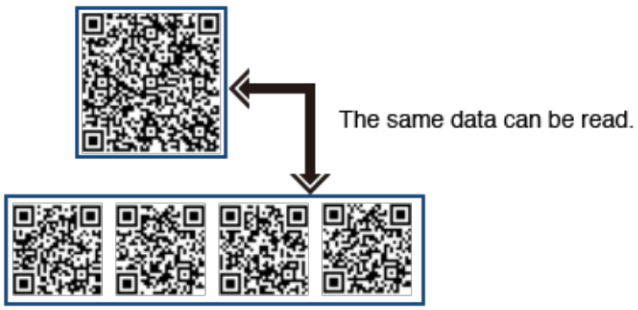
   QR *Code* dapat dibaca secara cepat dengan 360 derajat. Hal tersebut dikarenakan adanya pola deteksi posisi yang terletak pada tiga sudut simbol. Pola deteksi posisi ini menjamin pembacaan dengan kecepatan tinggi yang stabil, menghindari efek negatif dari gangguan latar belakang.



Gambar 2.25 Pola Deteksi Posisi QR *Code*

6. Fitur Penambahan Terstruktur

   QR *Code* dapat dibagi menjadi beberapa area data. Sebaliknya, informasi yang disimpan dalam beberapa simbol QR *Code* dapat direkonstruksi sebagai simbol data tunggal. Satu simbol data dapat dibagi hingga menjadi 16 simbol yang memungkinkan pencetakan di area yang sempit.



Gambar 2.26 Pembagian QR *Code*

2.7 *Bakery* dan *Patisserie*

Bakery dan *Patisserie* adalah dua aspek dari dapur *pastry* yang membuat perpaduan sempurna antara seni dan sains. Seni di *Bakery* dan *Patisserie* sebagian besar terbatas pada keterampilan presentasi dan pengaturan produk yang digunakan. Teknik dari sains mengidentifikasi bahan-bahan yang benar, memahami reaksi kimia pada setiap produk dan/atau kue yang dihasilkan, serta mengikuti proses yang tepat untuk membuat resep yang ideal. Karena kesadaran yang konstan akan kesehatan dan makan makanan organik, saat ini banyak profesional dan pembuat roti (*chef pastry*) yang sudah mulai unggul dalam memproduksi produk roti dan *pastry* bebas serat dan gluten tinggi untuk pelanggan yang cerdas [29].

Pada zaman dahulu, toko roti dibatasi untuk produksi roti dan biskuit, tetapi munculnya teknologi dan pengenalan bahan makanan baru seperti cokelat, gula, dan produk susu menciptakan serangkaian produk manis lain yang terkait dengan pembuatan roti, sehingga pada saat ini, *bakery* menghasilkan semua jenis produk seperti roti, *cakes*, kue kering, kue, *desserts*, dan sebagainya [29].

Bakery dan *Patisserie* telah menjadi salah satu karier paling dicari dalam bisnis saat ini. Banyak *chef* roti dan pembuat kue yang terampil juga mencari pekerjaan di pusat-pusat produksi roti, seperti *bakery*, *cafe*, hotel, dan sebagainya [29].

Terdapat beberapa jenis *pastry*, yaitu [30, 31]:

1. *Shortcrust Pastry*

   Jenis *pastry* ini merupakan pilihan yang paling populer untuk membuat kue *tart* dan *pie*. *Shortcrust pastry* diciptakan di Venesia dan resep pertama kali dicatat pada abad ke-18. Tekstur dari jenis *pastry* ini memiliki rasa mentega dan tekstur yang rapuh dan renyah.

2. *Phyllo pastry*

   Jenis *pastry* ini dibuat dalam lembaran yang sangat tipis dan digunakan sebagai selubung untuk banyak hidangan gurih dan manis yang lembut. Dibuat dengan tepung kandungan gluten tinggi, *phyllo* sangat sulit dibuat dan perlu penanganan yang hati-hati karena *pastry* tipis dan rapuh sehingga cepat kering.

3. *Puff Pastry*

   Puff pastry dibuat dengan bahan yang sama dengan *shortcrust pastry*. Lapisan adonan untuk *puff pastry* adalah kegiatan yang memakan waktu, tetapi hasilnya memuaskan. Air dan mentega yang mengembang menyebabkan jenis *pastry* ini naik dari uap dan menciptakan celah di antara lapisan-lapisan yang memberikan tekstur lapang.

4. *Rough Puff Pastry*

   Pastry jenis ini digunakan untuk gulungan sosis, lapisan *pie* gurih dan *tart*, serta lebih mudah dibuat daripada *puff pastry*.

5. *Choux Pastry*

   Choux pastry merupakan *pastry* asal Perancis yang cocok untuk diisi dengan krim pipi untuk membuat makanan/hidangan kecil yang lezat, seperti *éclair* dan *cream puffs*. Jenis *pastry* ini terbuat dari tepung, mentega, air, dan telur untuk kekayaan ekstra.

Ada pula yang termasuk dalam kategori *pastry*, yaitu [29]:

1. *Cookie*

   Cookie pada dasarnya adalah versi yang lebih kecil dan lebih kering dari kue. Tekstur *cookie* mungkin keras, rapuh, ringan, atau padat. Ada kombinasi rasa yang tak ada habisnya, karena banyak pilihan bahan dasar yang mencakup mentega, gula, telur, rasa, dan inklusi. Karena panduan tata rias dan *baking* sangat bervariasi antar *cookie*, pastikan untuk mengikuti instruksi dalam formula untuk teknik menghias dan waktu memanggang. Terlepas dari rasa yang berbeda, terdapat beberapa jenis *cookies*, yaitu *Dropped Cookie, Piped Cookie, Cut-Out Cookie, Sheet Cookie, Sliced Cookie, Icebox Cookie, Stencil Cookie,* dan*Molded Cookie*.

2. *Bread*

   Kategori *bread* (roti) mencakup berbagai produk seperti *muffin*, *scone*, biskuit, dan kue kopi. Keterampilan teknis yang terlibat dalam membuat roti terbatas pada

beberapa konsep utama yang mencakup fungsi bahan, metode pencampuran, dan proses memanggang.

3. *Pie*

   Sebagian besar dari apa yang diketahui tentang *pie* pada saat ini belum berubah sama sekali. Meskipun banyak yang menganggap kue ini berasal dari Amerika, ternyata *pie* berasal dari Eropa. *Pie* cenderung lebih populer di sekitar liburan musim dingin, dan beberapa toko roti hanya menawarkannya selama tahun itu. Tersedia dalam segala bentuk dan ukuran. Dua kategori utama *pie*, yaitu *pie* panggang dan *pie* yang tidak dipanggang.

4. *Cake*

   Dua kategori utama *cake* (kue) adalah berbasis lemak dan berbasis telur. Dalam kategori ini, beberapa jenis pencampuran hanya berlaku untuk satu *cake*, atau satu gaya *cake*.

2.8 *Certificate Pastry*

Sertifikat *pastry* merupakan sebuah sertifikat yang didapat dari mengikuti program yang memaparkan siswa pada kerajinan seni kue dan seni memanggang serta menekankan pengetahuan teoritis yang akan membantu siswa meraih kesuksesan menjadi seorang koki *pastry*. Program tersebut hanya menampilkan kursus yang terkait dengan seni *pastry* dan *baking*. Kursus-kursus tersebut antara lain, membuat makan penutup dan roti, kebersihan makanan dasar, dekorasi kue, teori memanggang, serta membuat roti yang tidak menggunakan mesin dan hanya menggunakan tangan (*artisan bread*) [32].

Berikut beberapa contoh sertifikat koki pastry:

Gambar 2.27 *Certificate Pastry*